SISTEMI OPERATIVI

ESERCIZIO N. 1 del 10 LUGLIO 2002

Un **servizio** del comune è fornito da **N+1 sportelli**. I **cittadini** che usufruiscono del servizio sono divisi in due categorie: **normali** e **anziani**. Il primo sportello dà sempre la priorità ai cittadini anziani, mentre per gli altri N non è definita una specifica politica di priorità. Dopo avere usufruito del servizio, è importante che ogni cittadino liberi esattamente lo sportello che aveva occupato.
Si implementi una soluzione usando il costrutto monitor per modellare gli sportelli e i processi per modellare i cittadini e si descriva la sincronizzazione tra i processi. Nella soluzione si massimizzi l'utilizzo delle risorse. Si discuta se la soluzione proposta può presentare starvation e in caso positivo per quali processi, e si propongano modifiche e/o aggiunte per evitare starvation.

```
program Comune

const     N = … { numero degli sportelli -1 }
type      tipo = (normale, anziano);  { tipo di cittadino }
type      dip = 0..N;  { numerazione degli sportelli }

type cittadino = process (t: tipo)
d: dip;
begin
      repeat
            d := s.richiedi (t);
            <usufruisci del servizio >
            s.rilascia (d);
      until false
end

type servizio = monitor

{ variabili del monitor }
var  libero : array[dip] of boolean;
      { stato di ogni dipendente }
      Doccupati : integer;
      { numero degli sportelli occupati }
      coda : array[tipo] of condition;
      { code su cui sospendere i cittadini }

function entry richiedi (t: tipo):dip
i: integer;
begin
      { se non ci sono dipendenti liberi }
      if Doccupati = N+1 then
            { sospensione }
            coda[t].wait;
      { occupo la risorsa }
      Doccupati++;
      for i := 0 to N do
```

```
            { cerco il primo dipendente libero }
            if libero[i]
            begin
                  libero[i] = false;
                  { ritorno il n. del dipendente occupato }
                  richiedi := i;
                  break;
            end
end

procedure entry rilascia (d: dip)
begin
      { rilascio le risorse }
      libero[dip] := true;
      Doccupati--;
      { se è il dipendente per gli anziani }
      if dip = 0
      begin
            { dà la precedenza agli anziani }
            if coda[anziano].queue then
                  coda[anziano].signal;
            else
                  coda[normale].signal;
      end
      else
      begin
            { dà la precedenza agli altri }
            if coda[normale].queue then
                  coda[normale].signal;
            else
                  coda[anziano].signal;
      end
end
```

```
begin { inizializzazione delle variabili }
      Doccupati := 0;
      for i := 0 to N do
            libero[i] = true;
end

var  s: servizio; { il nostro monitor }
      n1, n2, ... : cittadino (normale);
      a1, a2, ... : cittadino (anziano);

begin end.
```

Starvation

La soluzione proposta non presenta starvation perché comunque tutti i cittadini vengono prima o poi serviti. Potrebbe però presentare ritardi per il fatto che gli anziani vengono sempre scavalcati dai cittadini normali negli sportelli da 1 a N, e vengono serviti solo dal primo sportello.

Si può ridurre il ritardo imponendo un contatore per ogni tipo di cittadino, alternando la priorità ogni tot di accessi consecutivi dello stesso tipo.

NOTE

Il dipendente che dà priorità agli anziani è quello di indice 0. Non essendo definita una politica per gli sportelli 1..N, si è deciso di dare la priorità ai cittadini normali, ma potevano essere imposte anche altre politiche (ad es., alternare il risveglio tra i due tipi).